Squirrel: A decentralized peer-to-peer web cache

Paper by Sitaram Iyer, Antony Rowstron and Peter Druschel (© 2002)

Presentation* by Alexander Prohaska

*Slides partly taken from [2]

Little motivation for Squirrel

- Large organizations sometimes have clusters of more than 20 machines acting as web cache to handle peak loads
  - Expensive hardware and administration
- Decentralized peer-to-peer web cache
  - Every computer in the intranet takes a little part of the web caches job
  - Got the web cache for free!

What?

Overview

- Web caching
- Pastry
- Squirrel
- Evaluation
- Conclusion

Web caching

- Goals
- Principle
- Web cache
- Cooperative web cache
- Decentralized web cache
- Centralized vs. decentralized web cache

Goals of web caching

- Goals
  - Reduce latency
  - Decrease external traffic
  - Reduce load on web servers and routers

➔ Deployed at corporate network boundaries, ISPs, etc.

- Assumptions for corporate LAN
  - Located in a single geographical region
    - Latency between two LAN PCs << latency to external servers
    - Inner-LAN bandwidth >> external bandwidth

Principle of web caching

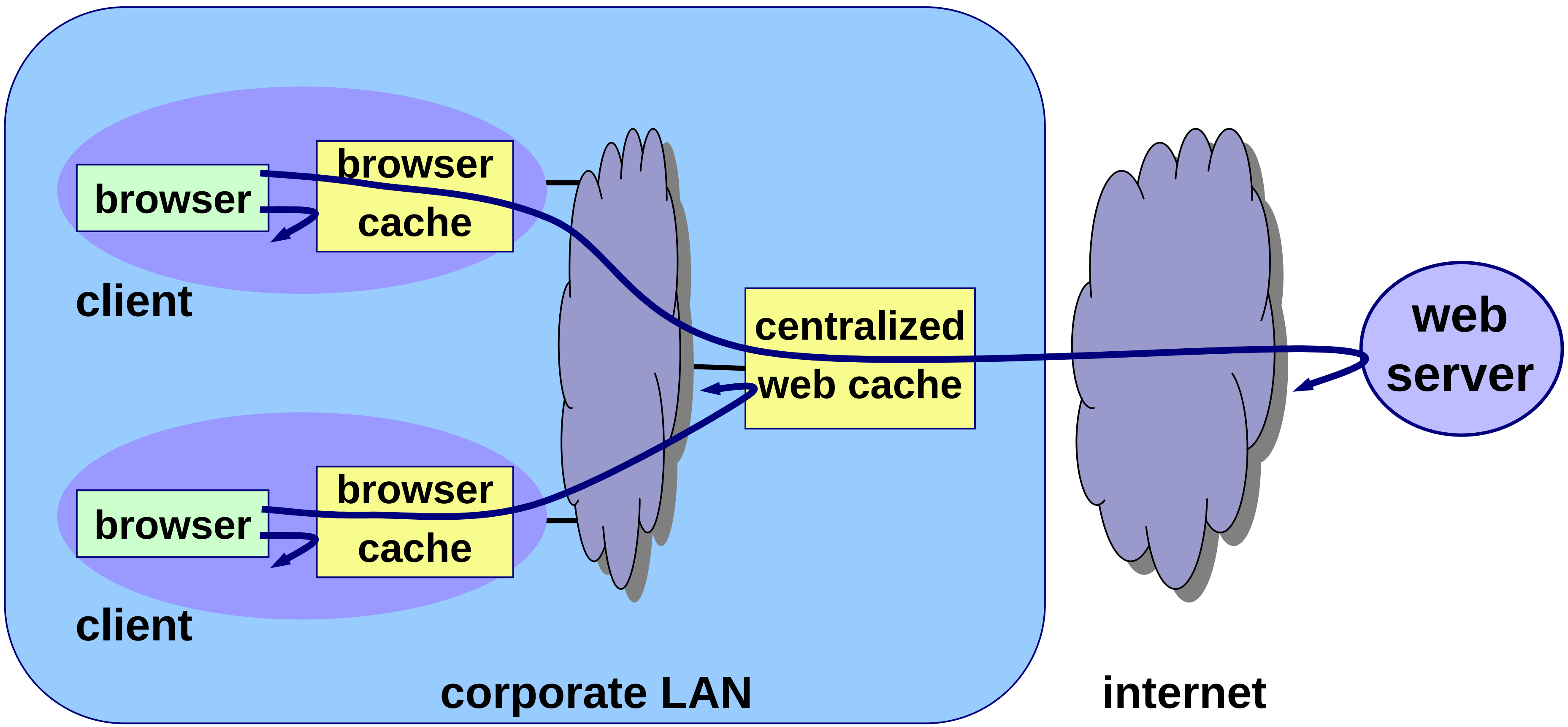
- Web browser makes HTTP GET requests for internet objects
  - Serviced from
    - local browser cache
    - web cache(s)
    - or origin web server

    depending on which has a fresh copy of the object

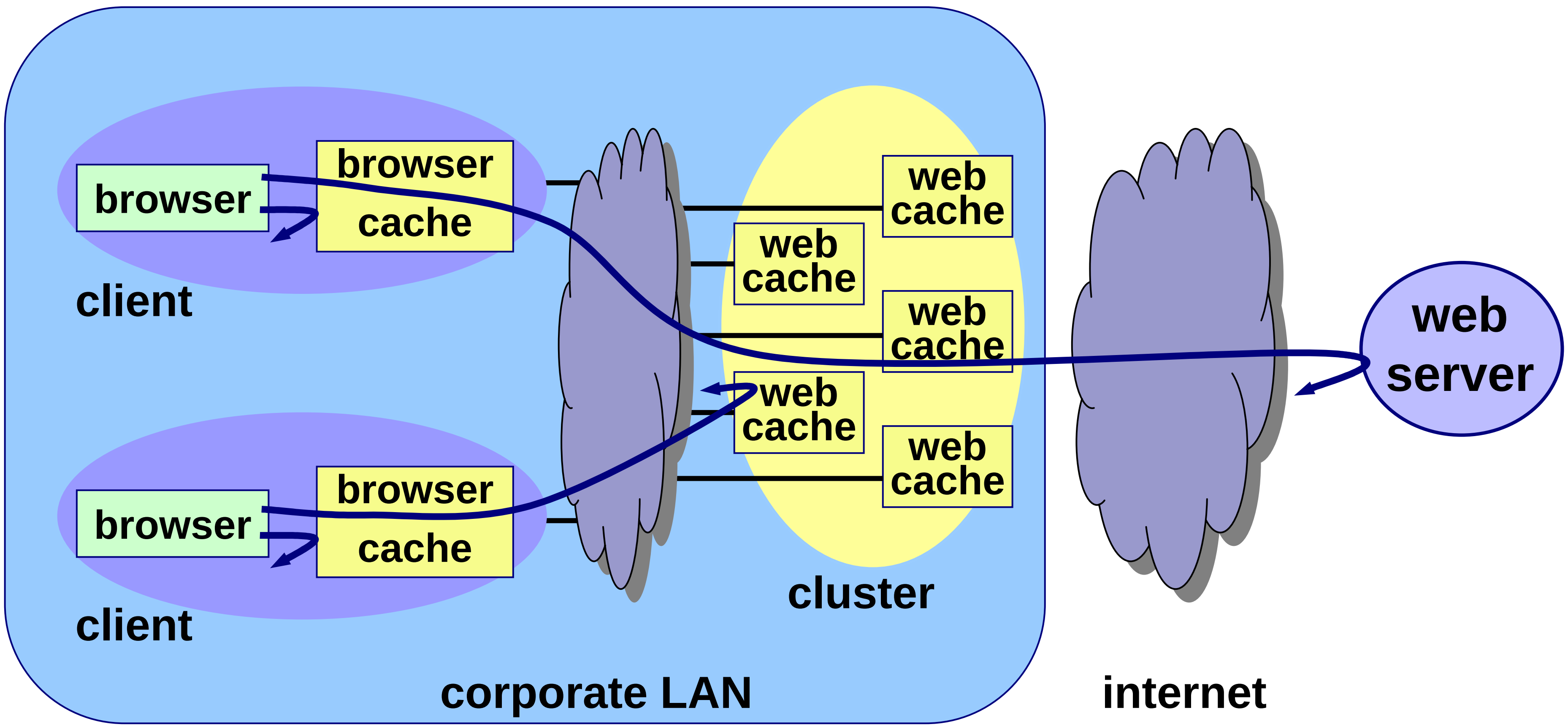
Principle of web caching (2)

- Object is not in local browser cache (cache miss or object is uncacheable)
  - Forward request to next level towards origin server
- Object is found in local browser cache
  - Check for freshness
    - If fresh, object is returned to the web browser,
    - otherwise local browser cache issues a *conditional GET* (*cGET*) request to the next level for validation
      - Typically an *If-Modified-Since* request with timestamp
      - Responses to cGET are the new object or "*not modified*"

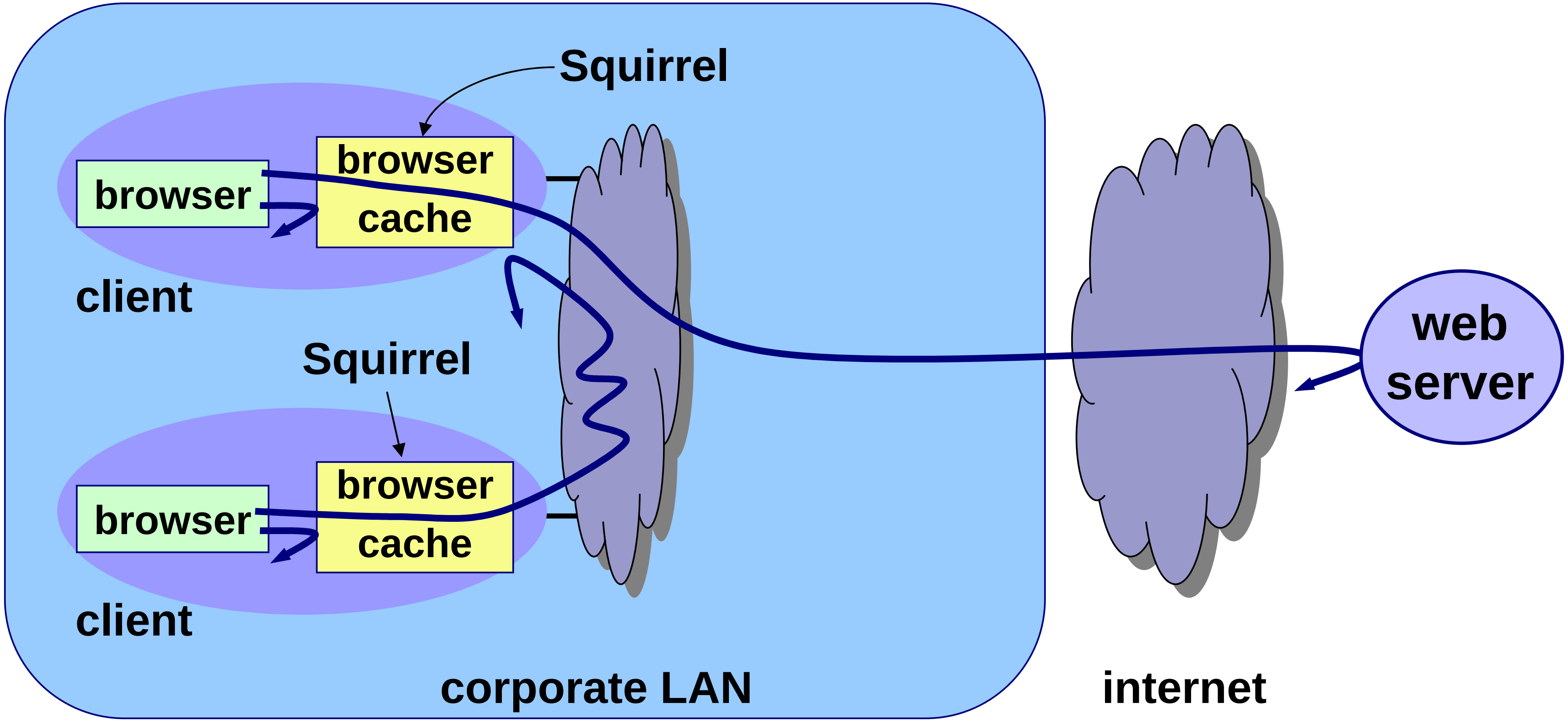
Web cache


browser
browser
cache
client
browser
browser
cache
client
centralized
web cache
web
server
corporate LAN
internet

Cooperative web cache



Decentralized web cache



Centralized vs. decentralized

Centralized web cache

- Administrative costs
- Dedicated hardware
- Scaling implies upgrading
- Single point of failure

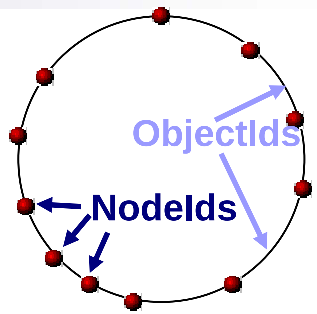
Decentralized (Squirrel)

- Self-organizing network
- No additional hardware
- Resources grow with clients
- Resilient of concurrent node failure

Overview

- Web caching
- Pastry
- Squirrel
- Evaluation
- Conclusion

Features of Pastry



- Self-organizing, decentralized & scalable DHT
- Circular 128-bit namespace for *NodeIds* and *ObjectIds* (uniformly, random)
- Objects are mapped to the live node numerically closest to ObjectIds
- A pastry node can route within $\mathbf{log_{2b}N}$ routing steps to the numerically closest node for a given ObjectId
- Automatically adapts to the arrival, departure and failure of nodes

Overview

- Web caching
- Pastry
- Squirrel
- Evaluation
- Conclusion

Squirrel

- Environment
- Scheme
- Mapping Squirrel onto Pastry
  - Home-store model
  - Directory model
  - Comparison

Environment of Squirrel

- 100 - 100.000 desktop machines in a corporate LAN
- Located in a single geographical region, e.g. a building or campus
- Each node
  - runs an instance of Squirrel,
  - disables the browsers cache and
  - sets Squirrel as the browser's proxy

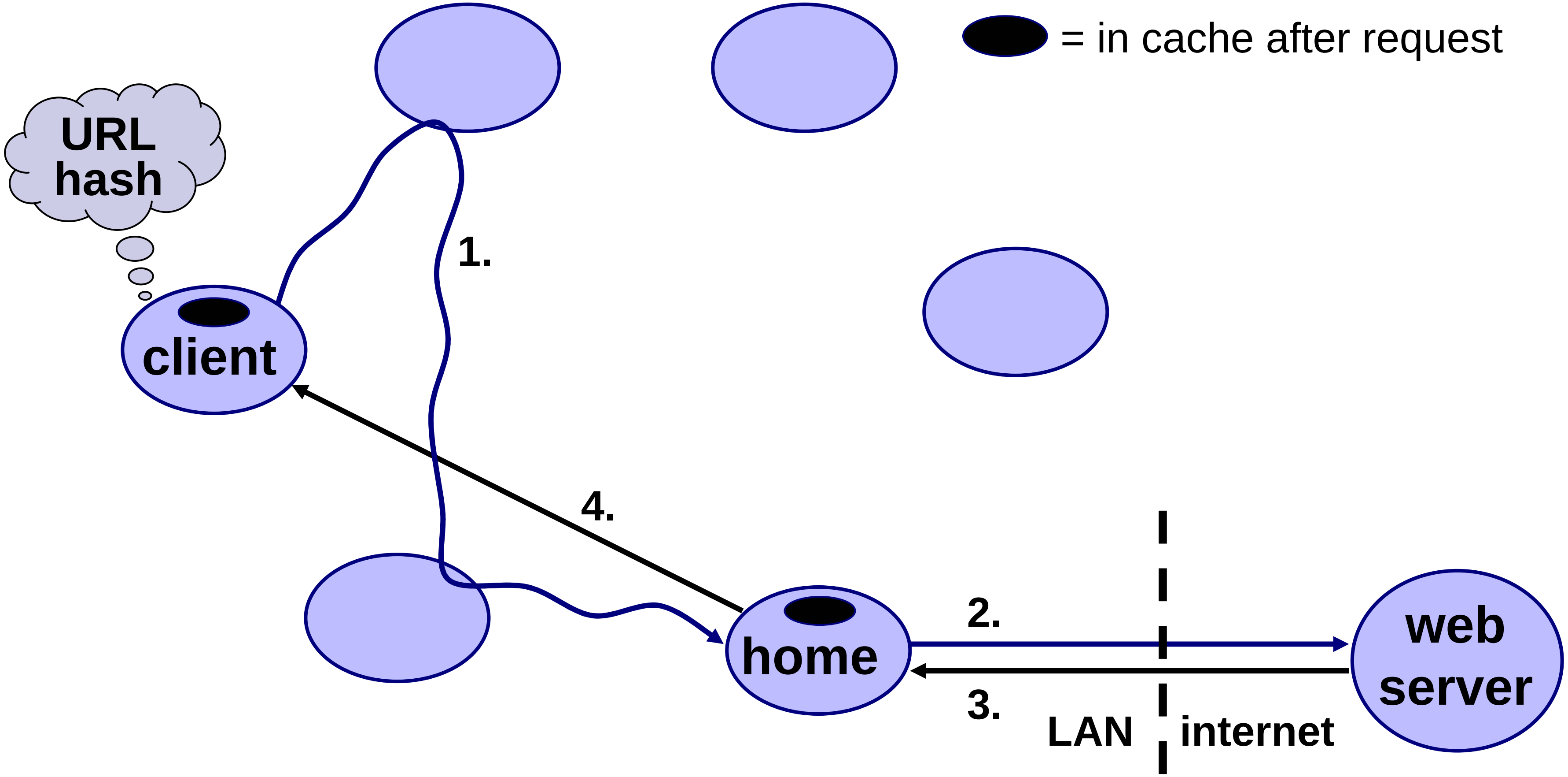
Scheme of Squirrel

- Web browser issues requests to Squirrel proxy
- Squirrel checks if the object is cacheable
- If it's cacheable and already in the cache, Squirrel checks freshness
- If it's not fresh, Squirrel maps the object to another Squirrel node using Pastry
- That's the *home node* for the object

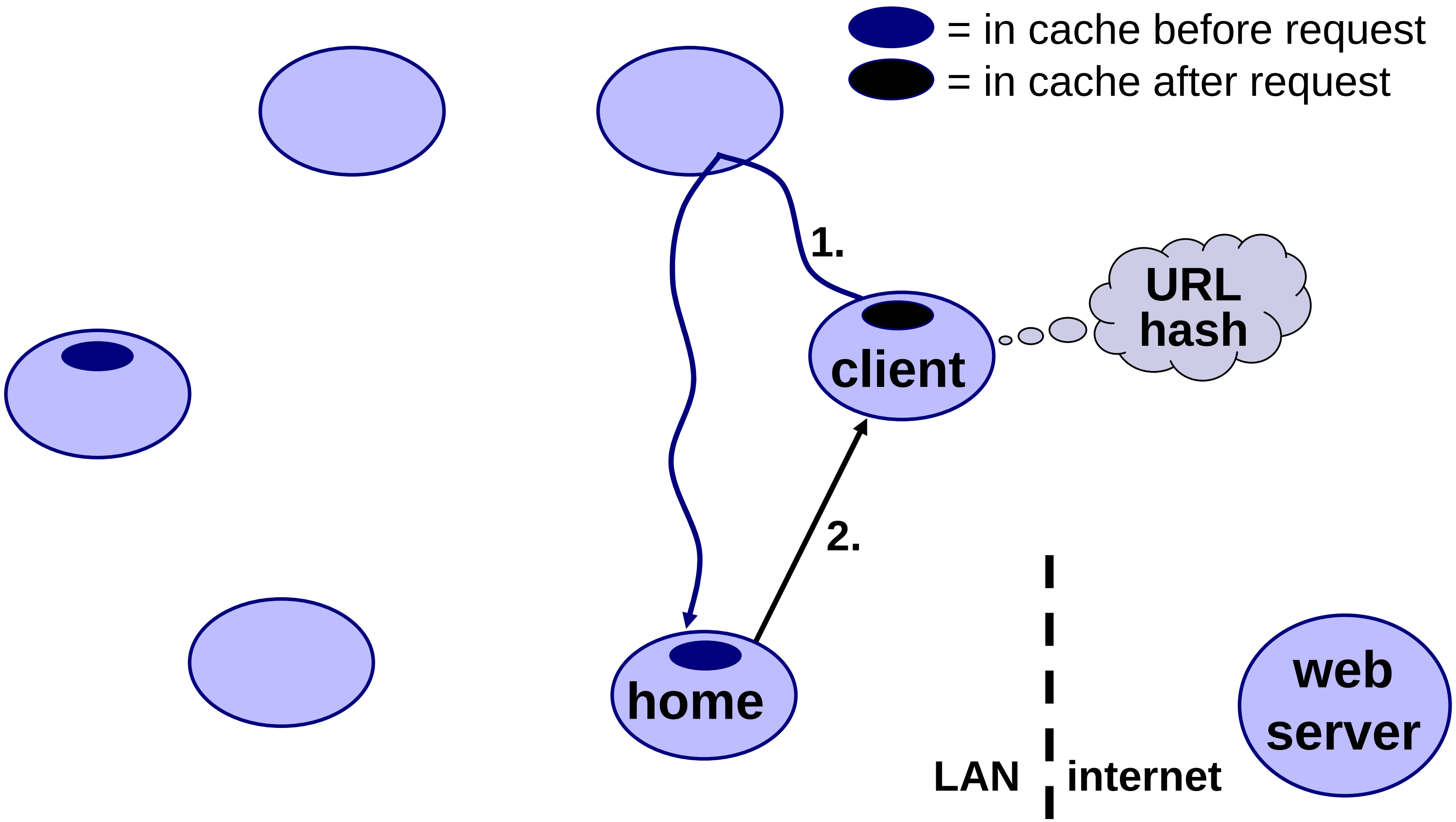
Mapping Squirrel onto Pastry

- Client nodes always cache objects locally
- Two approaches
  - Home-store model
    - Home node also stores object
  - Directory model
    - Home node remembers a small directory of (up to e.g. 4) pointers to recent clients (*delegates*)
    - Home node forwards subsequent requests to randomly chosen delegates with fresh object

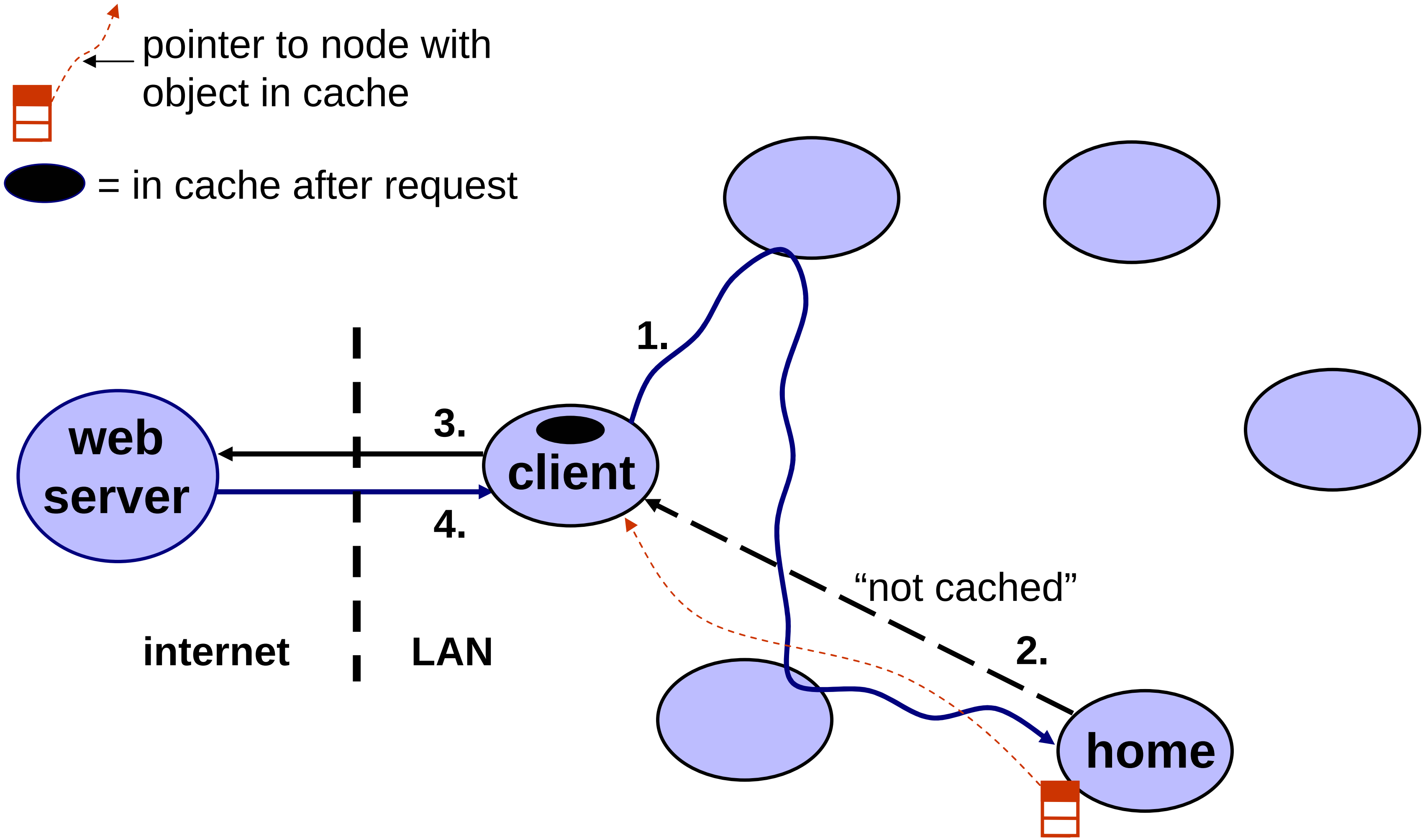
Home-store model (first request)



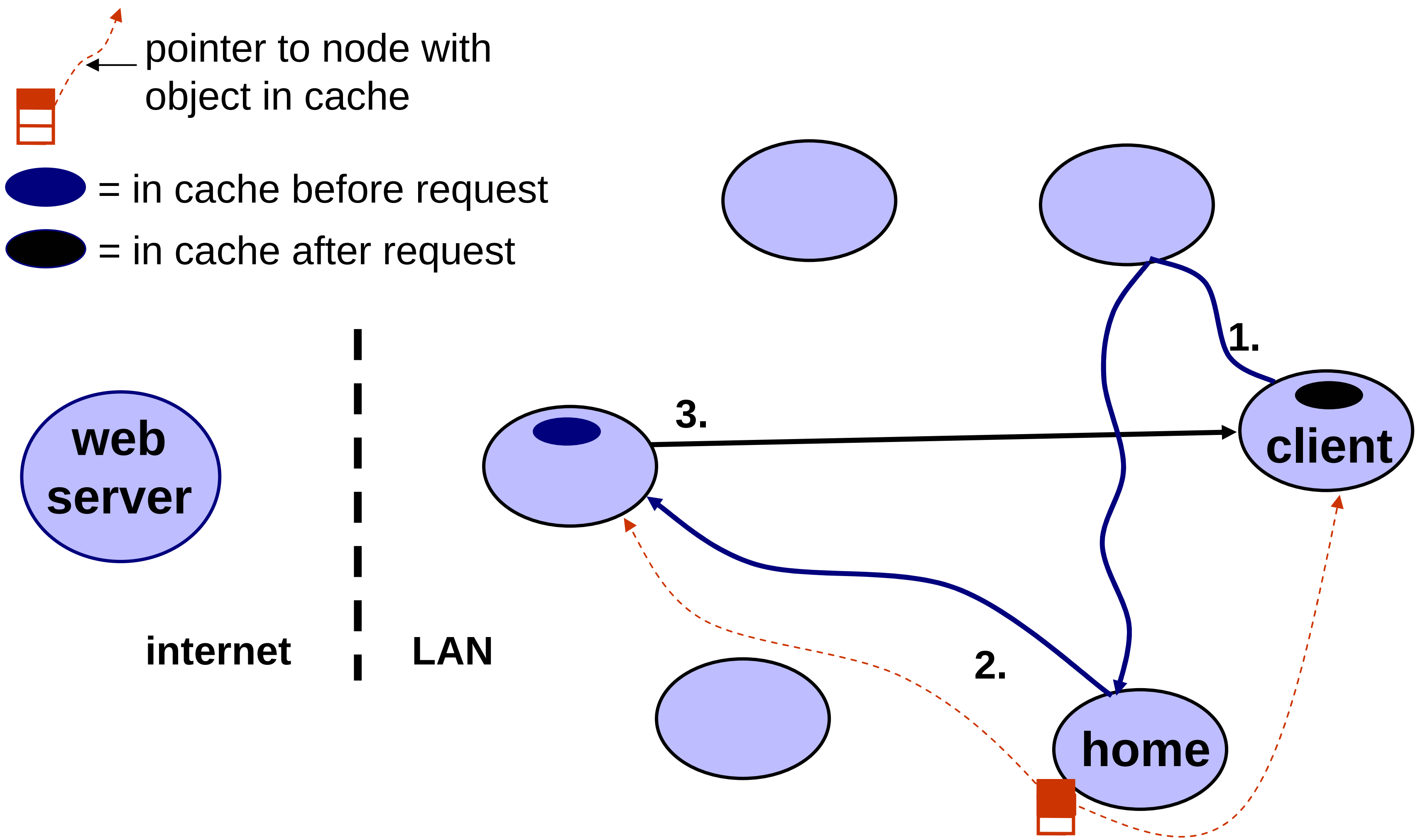
Home-store model (second request)



Directory model (first request)



Directory model (second request)



More on home-store model

- Advantages
  - It's simple
  - Hash function does mapping of objects to nodes, so popular files are uniformly distributed
- Disadvantages
  - Stores objects at home nodes and wastes storage
  - Copies of objects in the cache of non-home nodes are not shared in the network

More on directory model

- Advantages
  - Avoids storing unnecessary copies at the home node
  - Rapidly changing directory for popular objects seems to improve load balancing
- Disadvantages
  - Only requesting clients store objects in cache, so
    - active clients store all the popular objects and
    - inactive clients waste most of their storage
    - Seems to decline load balancing
- Improving or declining load balancing?

Overview

- Web caching
- Pastry
- Squirrel
- Evaluation
- Conclusion

Evaluation

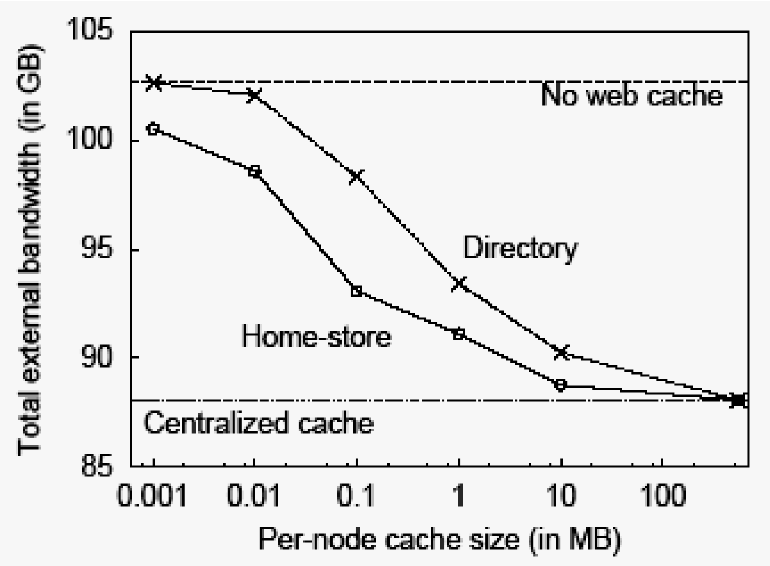
- Trace characteristics
- Total external traffic
- Latency
- Load
- Fault tolerance

Trace characteristics

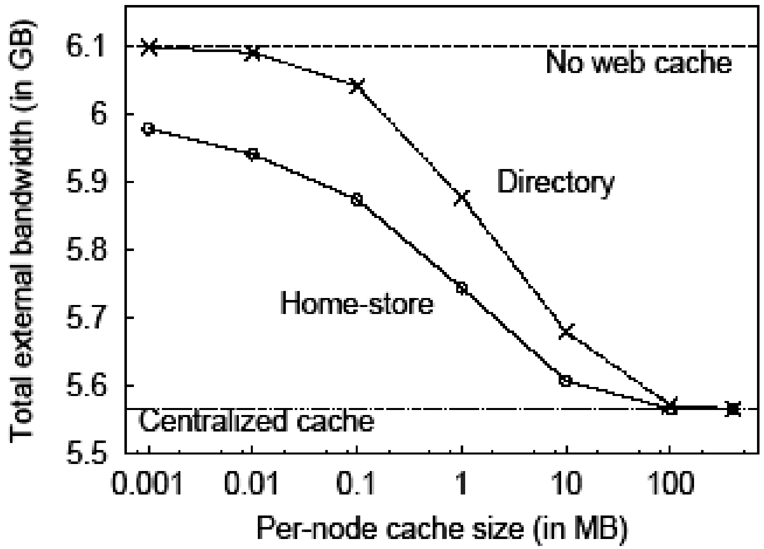
Microsoft located in :	Redmond	Cambridge
Total duration	1 day	31 days
Number of clients	**36.782**	**105**
Number of HTTP requests	16.41 million	0.971 million
Mean request rate	**190** req/s	**0.362** req/s
Peak request rate	**606** req/s	**186** req/s
Number of objects	5.13 million	0.469 million
Number of cacheable objects	2.56 million	0.226 million
Mean cacheable object reuse	5,4 times	3,22 times
Total external bandwidth	88,1GB	5,7GB
Hit ratio	29%	38%

Total external bandwidth

- The number of bytes transferred between Squirrel (LAN) and the internet (lower is better)



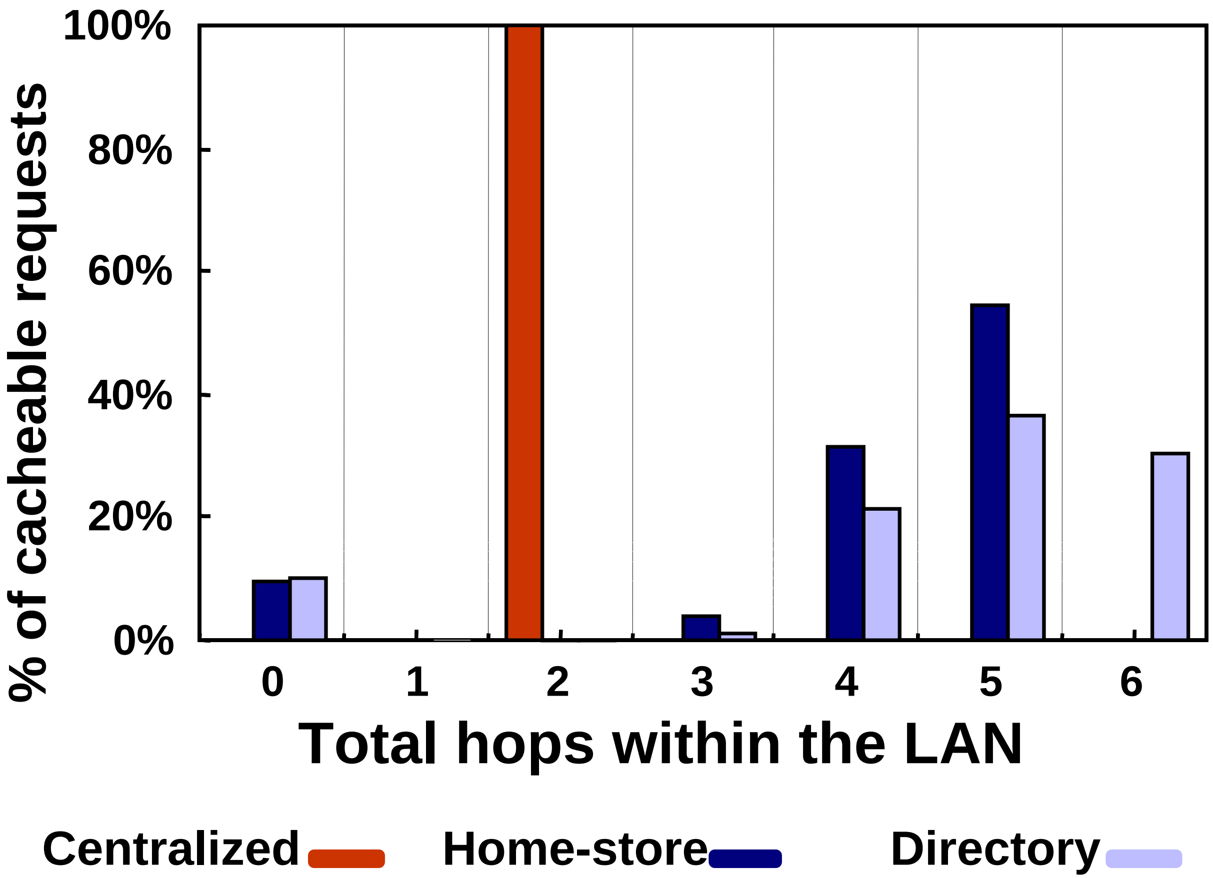
Redmond



Cambridge

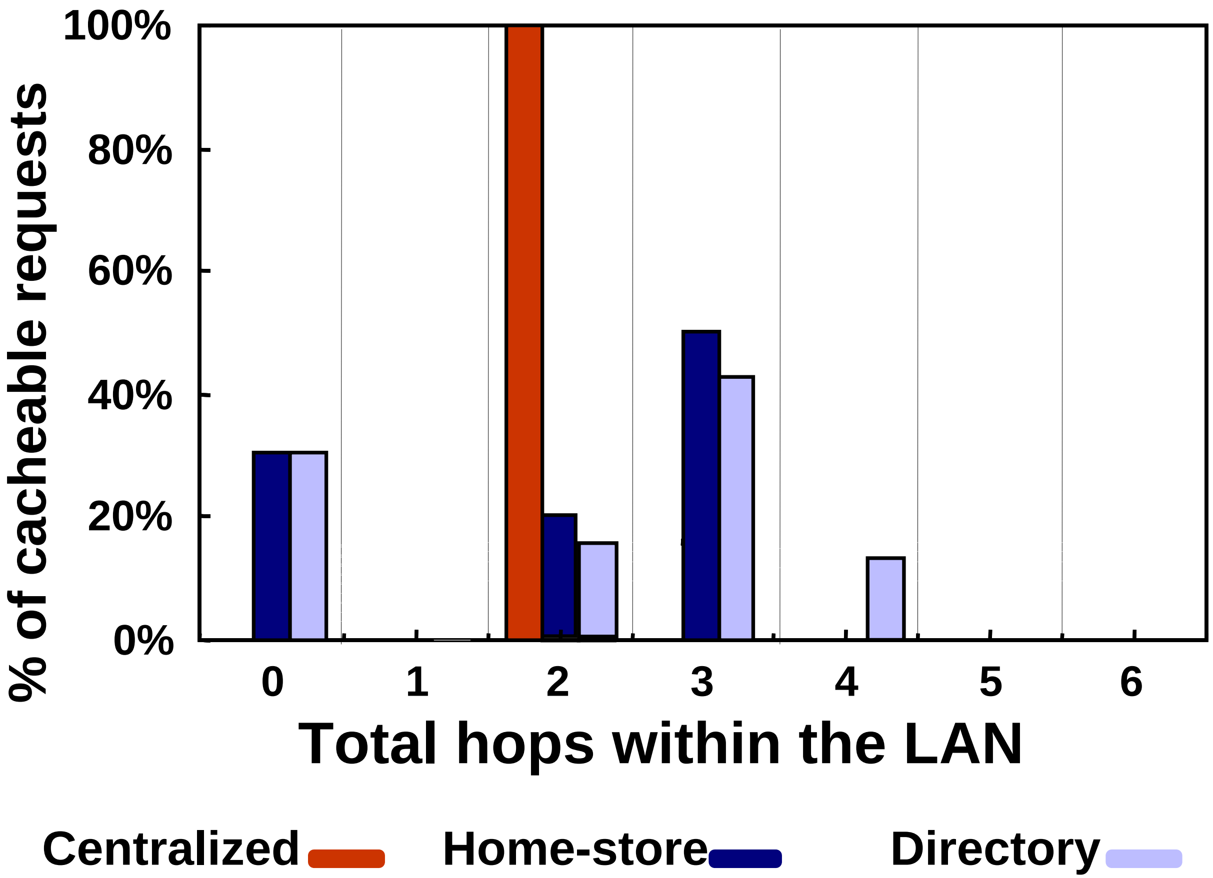
Latency (Redmond)

- Home-store needs between 3 and 4 hops to locate home node + 1 hop for the return path (mean 4,11 hops)
- Directory needs as many hops as home-store + 1 for forwarding to delegate (mean 4,56 hops)
- Additional, there are other cases



Latency (Cambridge)

- Like the Redmond trace, only shifted left by app. 2
- Home-store: mean 1,8 hops
- Directory : mean 2,0 hops



Peak load on single nodes

- Home-store model

Location	Objects/s	Objects/m	Av. objects/s	Av. objects/m
Redmond	8	65	1,5	36
Cambridge	9	35	0,038	1,13

- Directory model

Location	Objects/s	Objects/m	Av. objects/s	Av. objects/m
Redmond	48	388	6,6	60
Cambridge	55	125	0,027	0,7

Home-store model (hash function) causes drastically lower load compared to the directory model (access pattern)

Fault tolerance

- Sudden node failures result in partial loss of cached content
  - Home-store: proportional to failed nodes
  - Directory: more vulnerable
- If 1% of all Squirrel nodes abruptly crash, the fraction of lost cached content is:

	Home-store		Directory	
Redmond	Mean	1%	Mean	1.71%
	Max	1.77%	Max	19.3%
Cambridge	Mean	1%	Mean	1.65%
	Max	3.52%	Max	9.8%

Overview

- Web caching
- Pastry
- Squirrel
- Evaluation
- Conclusion

Conclusions

- Possible to decentralize web caching
- Performance comparable to a centralized web cache
- Better in terms of cost, scalability and administrative effort
- Under the made assumptions, the home-store scheme is superior to the directory scheme

References

[1] Sitaram Iyer, Antony Rowstron, Peter Druschel, Squirrel: A decentralized peer to peer web cache, July 2002, http://research.microsoft.com/~antr/PAST/squirrel.pdf

[2] Sitaram Iyer, Presentation slides from PODC, 21-24 July 2002, Monterey, CA., http://research.microsoft.com/PAST/squirrel-podc.pdf

[3] Antony Rowstron, Peter Druschel, Pastry: Scalable, decentralized object location and routing for large scale peer-to-peer systems, November 2001, http://research.microsoft.com/PAST/pastry.pdf

[4] Miguel Castro, Peter Druschel, Y. Charlie Hu, and Antony Rowstron, September 2002, Topology-aware routing in structured peer-to-peer overlay networks, http://www.ece.purdue.edu/~ychu/publications/fudico02.pdf

Finish

- Thanks for your attention
- Any questions?

Outtakes

- The following slides were taken out for timing reasons.

Pastry

- Features
- Node state
- Routing table
- Routing
- Routing example
- Self-organization

Pastry node state – an example

NodeId 10233102

Leaf set

SMALLER		LARGER	
10233033	10233021	10233120	10233122
10233001	10233000	10233230	10233232

Set of nodes with |L|/2 smaller and |L|/2 larger numerically closest NodeIds

Routing table

-0-2212102	1	-2-2301203	-3-1203203
0	1-1-301233	1-2-230203	1-3-021022
10-0-31203	10-1-32102	2	10-3-23302
102-0-0230	102-1-1302	102-2-2302	3
1023-0-322	1023-1-000	1023-2-121	3
10233-0-01	1	10233-2-32	
0		102331-2-0	
		2	

Prefix-based routing entries

Neighborhood set

13021022	10200230	11301233	31301233
02212102	22301203	31203203	33213321

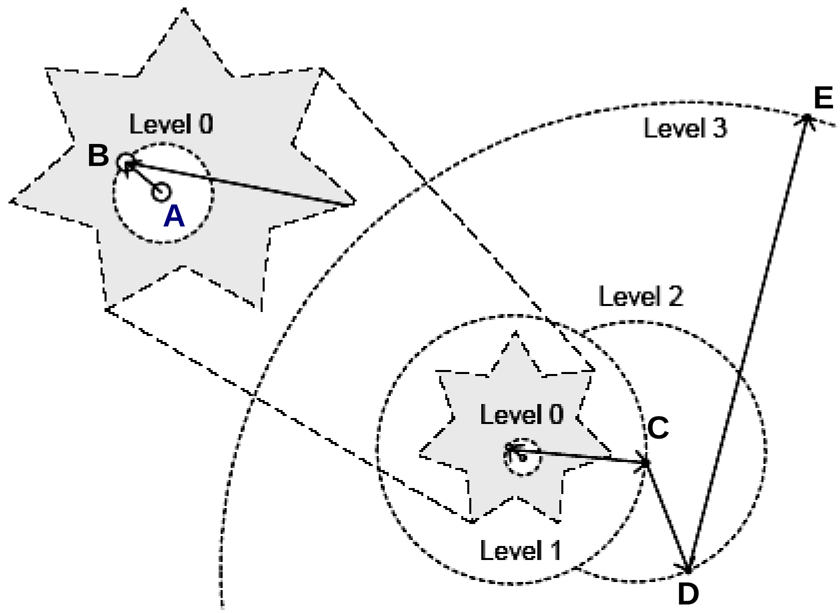
|M| “physically” closest nodes

Routing tables in Pastry

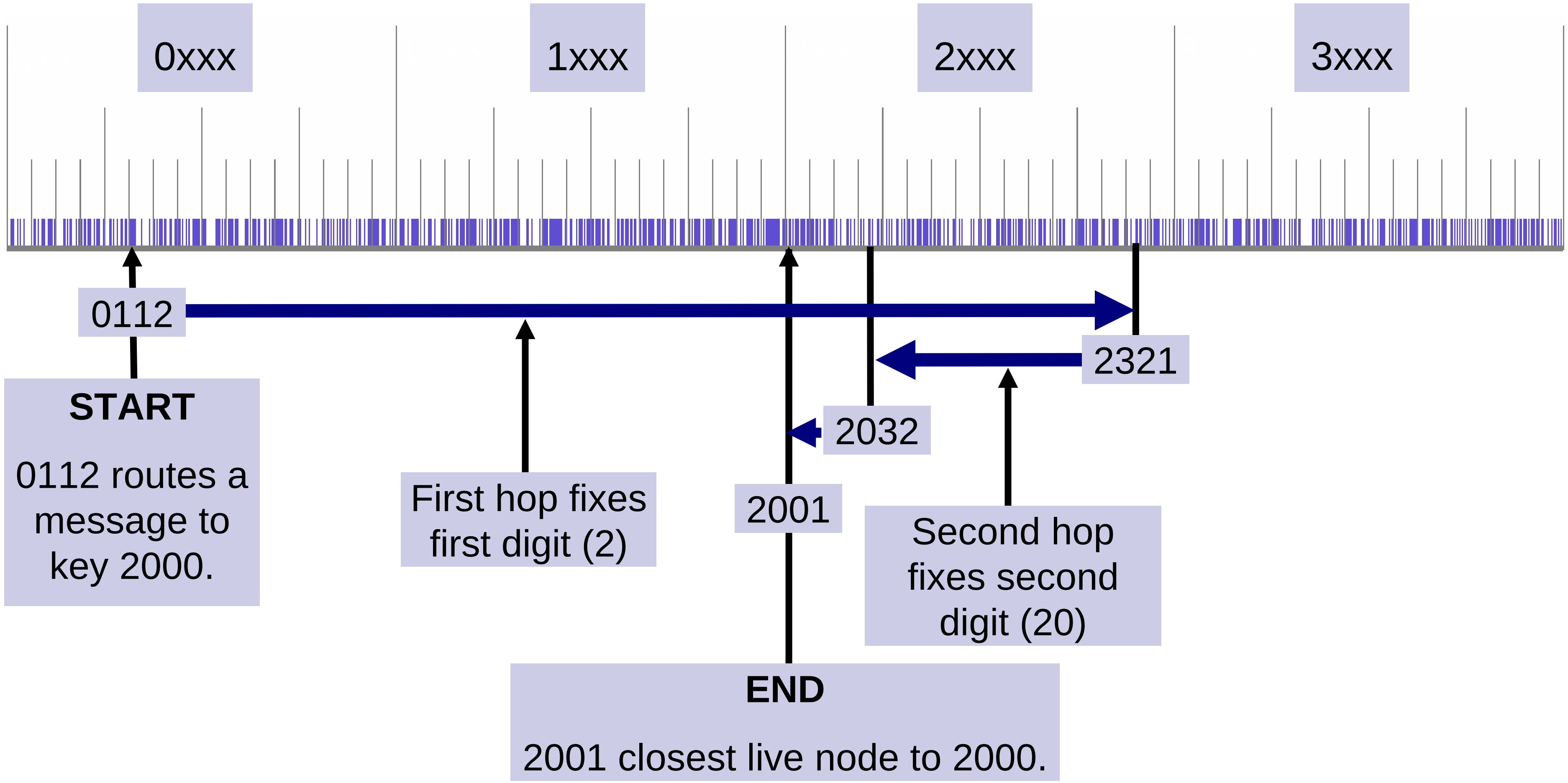
- NodeIds are in base 2^b
- One row for each prefix of local NodeId ($\log_{2^b} N$ populated on average)
- One column for each possible digit in the NodeId representation (2^b -1)
- Configuration parameter b defines the tradeoff:
  - ($\log_{2^b} N$) * (2^b -1) entries to route within
  - $\log_{2^b} N$ routing hops
  - Typical value for b is 4

Routing with Pastry

- No guarantee to find shortest path
- Gives rise to relatively good path
- Next node is chosen from an exponentially decreasing number of nodes
- Distance during each successive routing step is exponentially increasing
  - Dist(A,B)<Dist(B,C)
  - …



An example for routing in Pastry



Self-organization in Pastry

- Node arrival
  - Compute NodeId X (typically SHA-1 hash of IP address)
  - Find an nearby Pastry node automatically
  - Send special “join” message to X
    - Pastry will route this message to the existing node numerically closest to the new node
    - All nodes that route or receive the message update their state table and send it to X
    - Build up own state table by taking the appropriate parts of the received state tables

Self-organization in Pastry (2)

- Node departure
  - Nodes may fail or depart without warning
  - Pastry guarantees that each node lazily repairs its leaf set unless |L|/2 nodes with adjacent NodeIds have failed simultaneously
    - L is the second configuration parameter, typical value is between 16 and 32
  - Inaccurate routing table entries must be replaced to preserve the integrity of routing tables
    - Meanwhile, messages are routed over a node that is numerically closer to the destination than the current node

More on directory model (2)

- Load spike example
  - Requests for web pages with many embedded images will all be served by a few recent clients
  - Many home nodes point to such clients
- Declining or improving load balance?
  - Evaluation will show

Hit Ratio

- Defined as the fraction of all objects that are serviced by the web cache
- Hit ratio of the Squirrel cache is indirectly related to the external bandwidth
- Squirrel approaches hit ratio of centralized cache with increasing per-node contribution
- At about 100 MB per node cache size, Squirrel achieves 28% and 37% for Redmond and Cambridge traces (out of possible 29% and 38%), respectively